Three-dimensional Geometric Transformations :-

Each geometric transformation operator is now 4x4 matrix, which premultiplies a coordinate column vector.

- Three-dimensional translation
- Three-dimensional rotation
- Three-dimensional scaling

1. Three-dimensional translation :-

$$P' = T \cdot P$$

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & tx \\ 0 & 1 & 0 & ty \\ 0 & 0 & 1 & tz \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

2. Three-dimensional rotation :-

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

→ Transformation equations for rotations about the other two coordinate axes can be obtained with a cyclic permutation of the coordinate parameters x, y and z in Equations:

$$x' = x\cos\theta - y\sin\theta$$
$$y' = x\sin\theta + y\cos\theta$$
$$z' = z$$

$$x \to y \to z \to x$$

Thus, to obtain the x-axis and y-axis rotation transformations, we cyclically replace x with y, y with z, and z with x.

for x-axis rotation:

$$y' = y\cos\theta - z\sin\theta$$
$$z' = y\sin\theta + z\cos\theta$$
$$x' = x$$

for y-axis rotation,

$$z' = z\cos\theta - x\sin\theta$$
$$x' = z\sin\theta + x\cos\theta$$
$$y' = y$$

→ In the special case where an object is to be rotated about an axis that is parallel to one of the coordinate axes, we attain the desired rotation with the following transformation sequence:

1. Translate the object so that the rotation axis coincides with the parallel coordinate axis.
2. Perform the specified rotation about that axis.
3. Translate the object so that the rotation axis is moved back to its original position.

→ Three-dimensional scaling :-

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

* Quaternion Methods for 3-D Rotations:

A more efficient method for generating a rotation about an arbitrarily selected axis is to use a quaternion representation for the rotation transformation.

→ Quaternions, which are extensions of two-dimensional complex numbers, are useful in a number of computer-graphics procedures, including the generation of fractal objects.

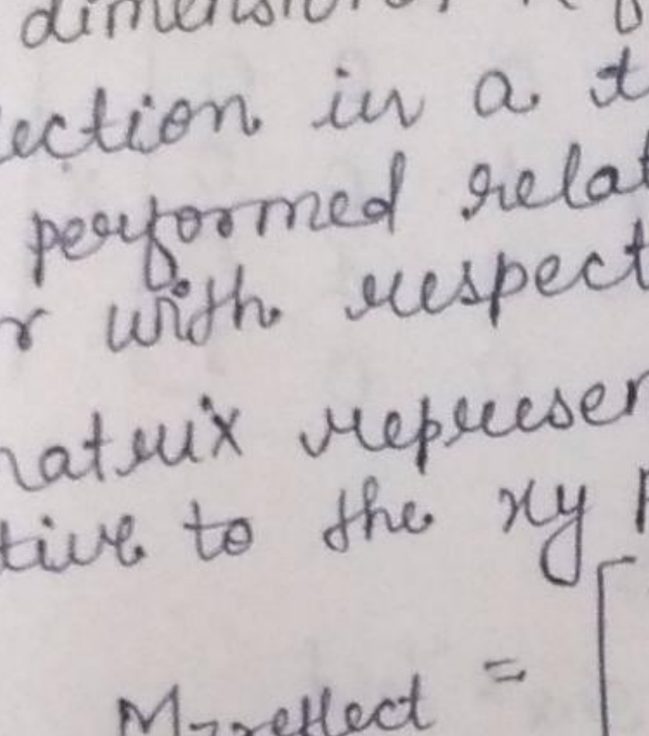
→ Composite Three-dimensional Transformations :-

→ Three-dimensional Reflections :-

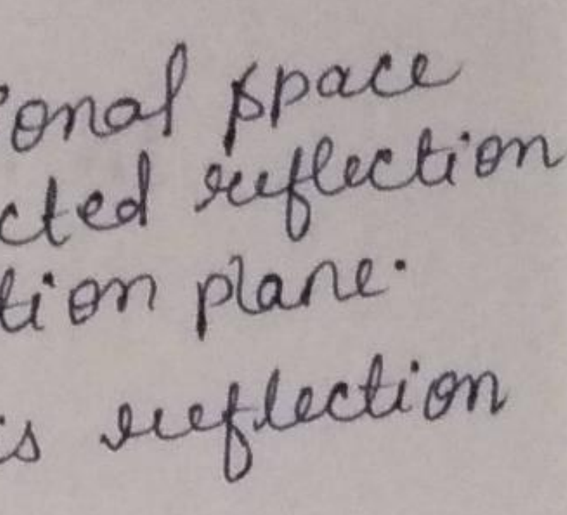
A reflection in a three-dimensional space can be performed relative to a selected reflection axis or with respect to a reflection plane.

The matrix representation for this reflection relative to the xy plane is :

$$M_{z,reflect} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



Reflection relative to the xy plane



→ Three-dimensional Shear :-

A general z-axis shearing transformation relative to a selected reference position is produced with the following matrix:

$$M_{z,shear} = \begin{bmatrix} 1 & 0 & sh_{zx} & -sh_{zx} \cdot z_{ref} \\ 0 & 1 & sh_{zy} & -sh_{zy} \cdot z_{ref} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

→ Three-dimensional shear :-

→ Shearing in X axis:-

$$X_{new} = X_{old}$$
$$Y_{new} = Y_{old} + Sh_y * X_{old}$$
$$Z_{new} = Z_{old} + Sh_z * X_{old}$$

In matrix form,

$$\begin{bmatrix} X_{new} \\ Y_{new} \\ Z_{new} \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ Sh_y & 1 & 0 & 0 \\ Sh_z & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} X_{old} \\ Y_{old} \\ Z_{old} \\ 1 \end{bmatrix}$$

→ Shearing in Y Axis

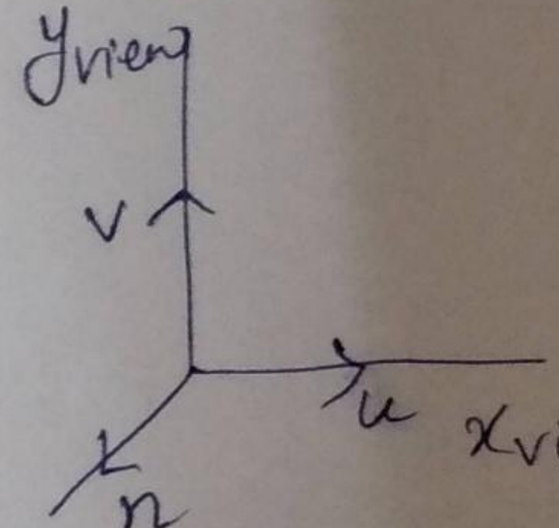
$$X_{new} = X_{old} + Sh_x * Y_{old}$$
$$Y_{new} = Y_{old}$$
$$Z_{new} = Z_{old} + Sh_z * Y_{old}$$

→ Shearing in Z Axis:

$$X_{new} = X_{old} + Sh_x * Z_{old}$$
$$Y_{new} = Y_{old} + Sh_y * Z_{old}$$
$$Z_{new} = Z_{old}$$

* The uvn Viewing-coordinate Reference Frame

- Left-handed viewing coordinates are sometimes used in graphics packages, with the viewing direction in the positive zview direction.
- With a left-handed system, increasing zview values are interpreted as being farther from the viewing position along the line of sight.
- But right-handed viewing systems are more common, because they have the same orientation as the world-ref. frame.
- The vector cross product of N and V also produces the adjusted value for V, perpendicular to both N and V, along the +ve yview axis.



$$n = \frac{N}{|N|} = (n_x, n_y, n_z)$$

$$u = \frac{V \times n}{|V \times n|} = (u_x, u_y, u_z)$$

$$v = n \times u = (v_x, v_y, v_z)$$

Viewing concepts

→ Viewing functions process the object descriptions through a set of procedures that ultimately project a specified view of the objects onto the surface of a display device.

→ Many processes in 3-D viewing, such as the clipping routines, are similar to those in the 2-D viewing pipeline.

→ But 3-D viewing involves some tasks that are not present in 2-D viewing.
for e.g: Projection routines are needed to transfer the scene to a view on a planar surface, visible parts of a scene must be identified, and, for a realistic display, lighting effects and surface characteristics must be taken into account.

→ Three-dimensional Viewing:

(i) Viewing a 3-D scene
(ii) Projections
(iii) Depth-Cueing
(iv) Identifying Visible Lines and Surfaces
(v) Surface Rendering
(vi) Exploded and Cutaway Views
(vii) Three-dimensional and stereoscopic Viewing

** Three-dimensional viewing :-

① Viewing a 3-D scene

- To obtain a display of a 3D world-coordinate scene, we first set up a coordinate reference for the viewing, or "camera", parameters.
- This coordinate ref defines the position & orientation for a view plane (or projection plane) that corresponds to a camera film plane
- Object descriptions are then transferred to the viewing reference co-ordinates and projected onto the view plane.

* ② Projections

- We can choose different methods for projecting a scene onto the view plane.

(i) One method for getting the description of a solid object onto a view plane is to project points on the object surface along parallel lines. This technique is called parallel projection.

(a) → It is used in engineering and architectural drawings to represent an object with a set of views that show accurate dimensions of the object.

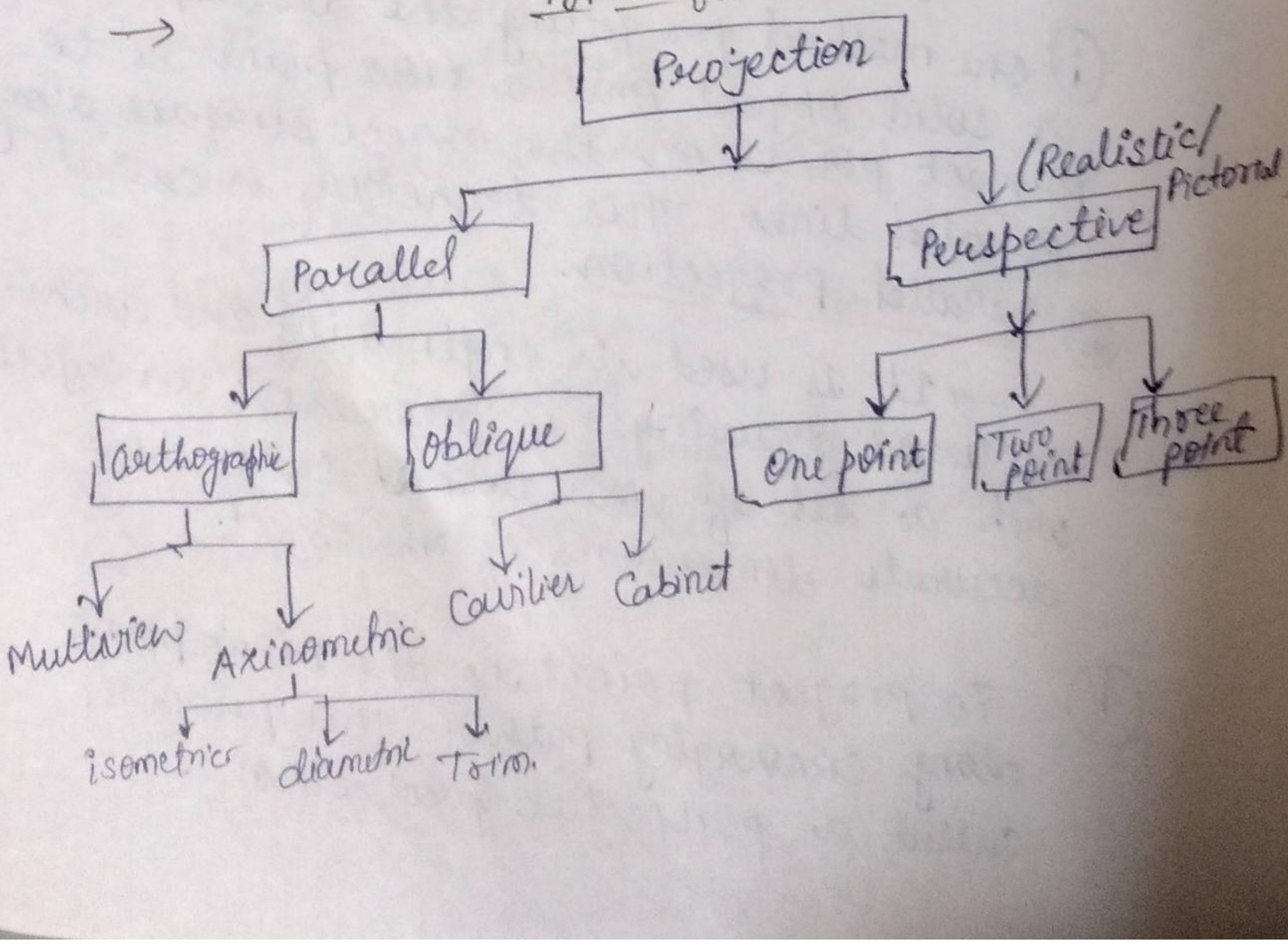
(ii) To project points to the view plane along converging paths. This process, called a perspective projection.

- It causes objects farther from the viewing position to be displayed smaller than objects of the same size that are nearer to the viewing position.

Note:-

- A scene that is generated using a perpective projection appears more realistic, because this is the way that our eyes and a camera lens form images.
- Parallel lines along the viewing direction appear to converge to a distant point in the background, and objects in the background appear to be smaller than objects in the foreground.

Types of Projection



** (iii) Depth cueing

→ Depth information is important in a three-dimensional scene so that we can easily identify, for a particular viewing direction, which is the front and which is the back of each displayed object.

→ A simple method for indicating depth with wire-frame displays is to vary the brightness of line segments according to their distances from the viewing position.

→ Depth cueing is applied by choosing a maximum and a minimum intensity value and a range of distances over which the intensity is to vary.

** 3-D Rotation :-

~~P'Z = RP~~

$x \to y \to z \to x$ ⊙ z-Rotation

Row major.

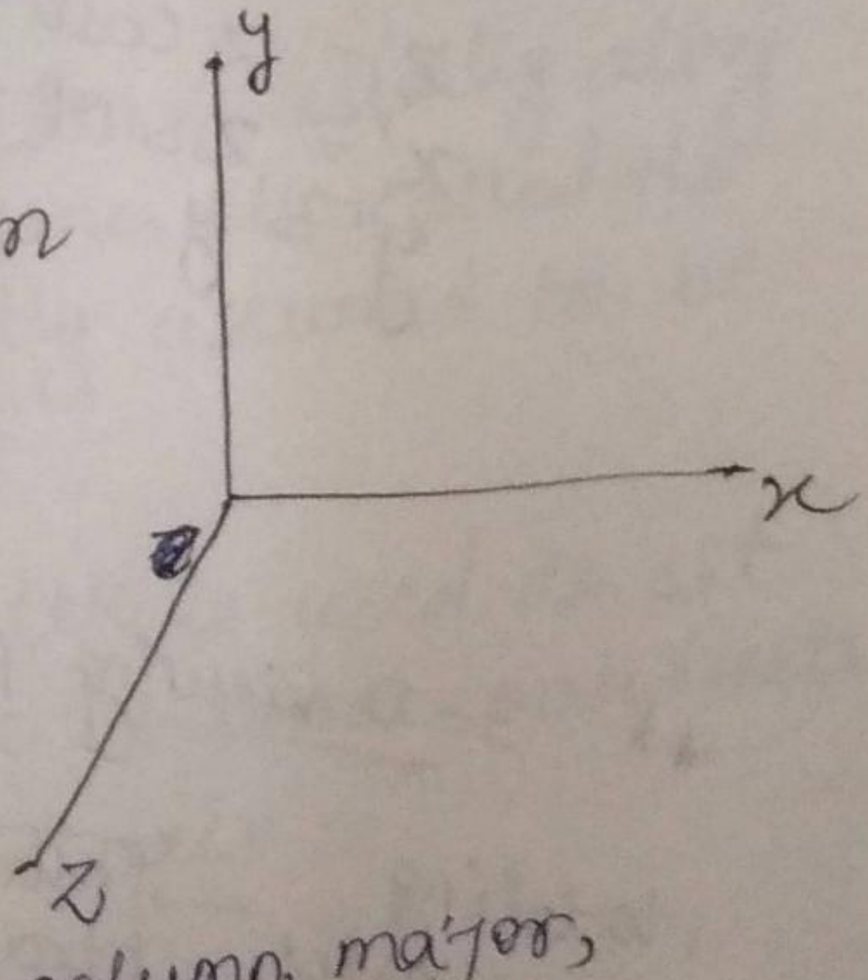
$P' = P \cdot R$

$x' = x\cos\theta - y\sin\theta$

$y' = x\sin\theta + y\cos\theta$

$z' = z$

$$R = \begin{array}{c} \\ x \\ y \\ z \\ 1 \end{array} \begin{array}{c} \begin{array}{cccc} x' & y' & z' & 1 \end{array} \\ \begin{bmatrix} \cos\theta & \sin\theta & 0 & 0 \\ -\sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \end{array}$$



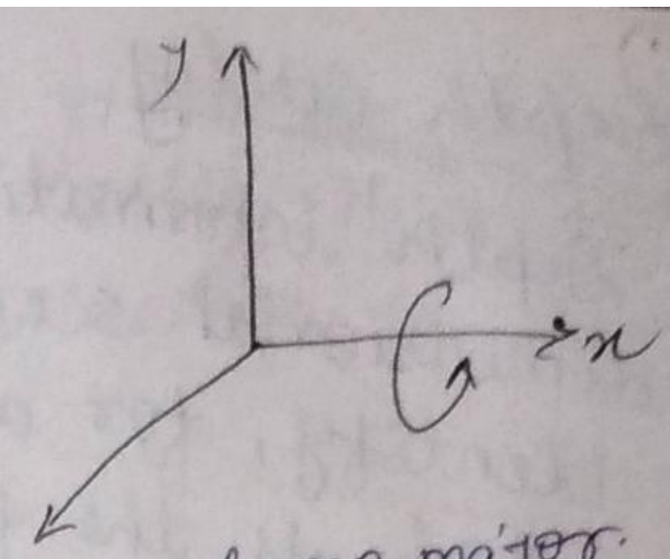
column major,

$P' = R \cdot P$

$$R = \begin{array}{c} \\ x' \\ y' \\ z' \\ 1 \end{array} \begin{array}{c} \begin{array}{cccc} x & y & z & 1 \end{array} \\ \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \end{array}$$

= x axis Rotation.

$y' = y\cos\theta - z\sin\theta$
$z' = y\sin\theta + z\cos\theta$
$x' = x$



Row major

$$R_x(\theta) = \begin{array}{c} \\ x \\ y \\ z \\ 1 \end{array} \begin{array}{c} \begin{array}{cccc} x' & y' & z' & 1 \end{array} \\ \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & \sin\theta & 0 \\ 0 & -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \end{array}$$

Column major.

$$R_x(\theta) = \begin{array}{c} \\ x' \\ y' \\ z' \\ 1 \end{array} \begin{array}{c} \begin{array}{cccc} x & y & z & 1 \end{array} \\ \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \end{array}$$

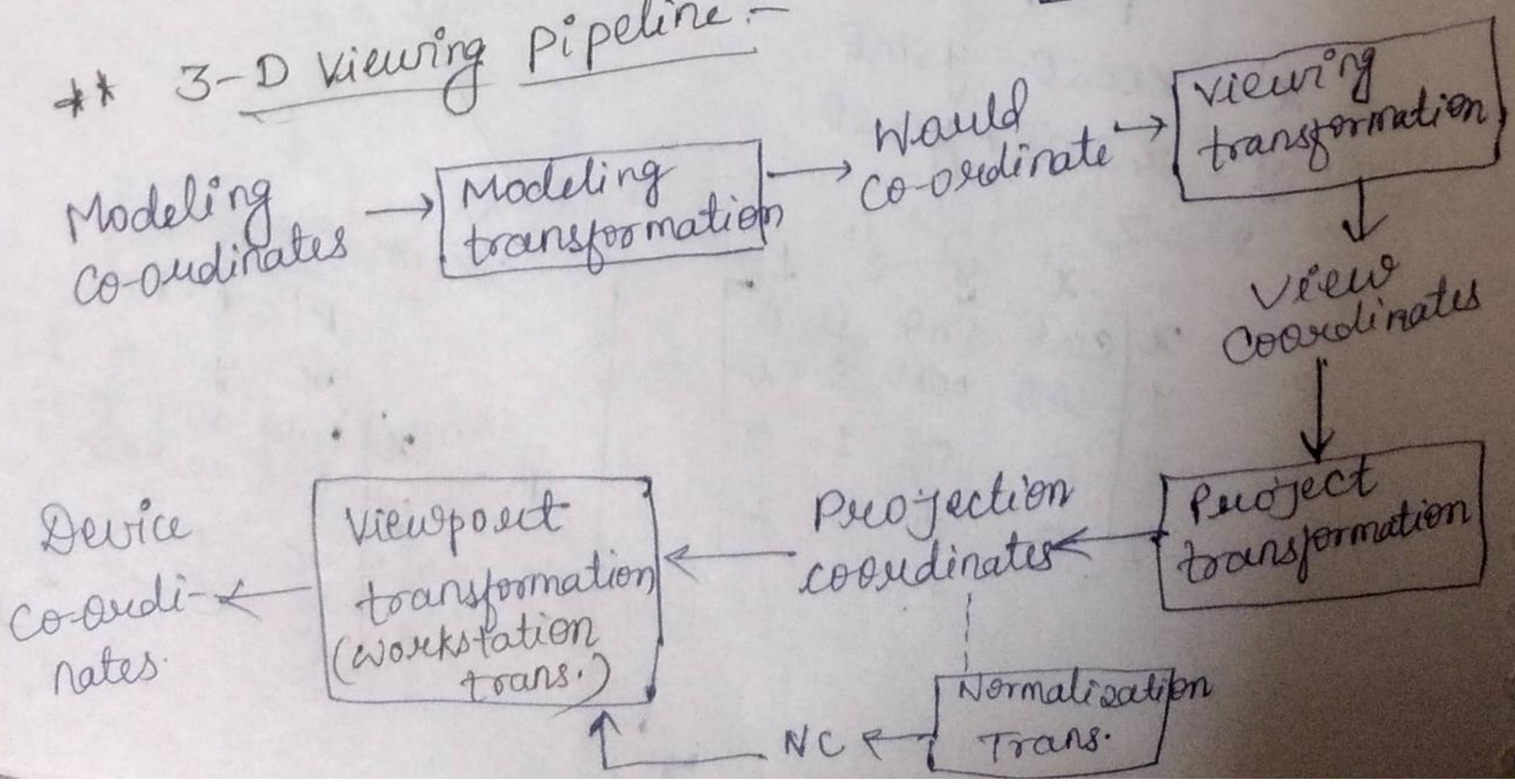
= y axis Rotation

$z' = z\cos\theta - x\sin\theta$
$x' = z\sin\theta + x\cos\theta$
$y' = y.$

Column Major

$$R_y(\theta) = \begin{array}{c} \\ x' \\ y' \\ z' \\ 1 \end{array} \begin{array}{c} \begin{array}{cccc} x & y & z & 1 \end{array} \\ \begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ \sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \end{array}$$

** 3-D Viewing Pipeline :-



3D-Viewing parameter (how 3D object displays at 2D)

MC → Modeling transformation → WC → Viewing transformation → VC → Projection transformation → PC ← Viewport trans. ← DC

- Establishing a three-dimensional viewing reference frame is similar to setting up the two-dimensional viewing reference frame
→ We first ~~set~~ select a would-coordinate position $P_0 = (x_0, y_0, z_0)$ for the viewing origin, which is called the view point or viewing position.
- And we specify a view-up vector V, which defines the yview direction.
- Because the viewing direction is usually along the zview axis, the view plane, also called the projection plane, is normally assumed to be perpendicular to this axis.
- An additional scalar parameter is used to set the position of the view plane at some coordinate value z_{vp} along the zview axis.

** The View-up Vector
↳ defines the yview direction.

3-D Viewing Pipeline:-

→ (i) Transformation from World to Viewing Coordinates :—

— In the 3-D viewing pipeline, the first step after a scene has been constructed is to transfer object descriptions to the viewing-coordinate reference frame.

— This conversion of object descriptions is equivalent to a sequence of transformations that superimposes the viewing reference frame onto the world frame.

— (1.) Translate the viewing-coordinate origin to the origin of the world-coordinate system.

(2.) Apply rotations to align the xview, yview and zview axes with the world xw, yw and zw axes, resp.

** PROJECTION (ii) Projection Transformations

— Projection is a process by which we can create an image of an object on a plane.

— In the next phase of the 3-D viewing pipeline, after the transformation to viewing coordinates, object descriptions are projected to the view plane.

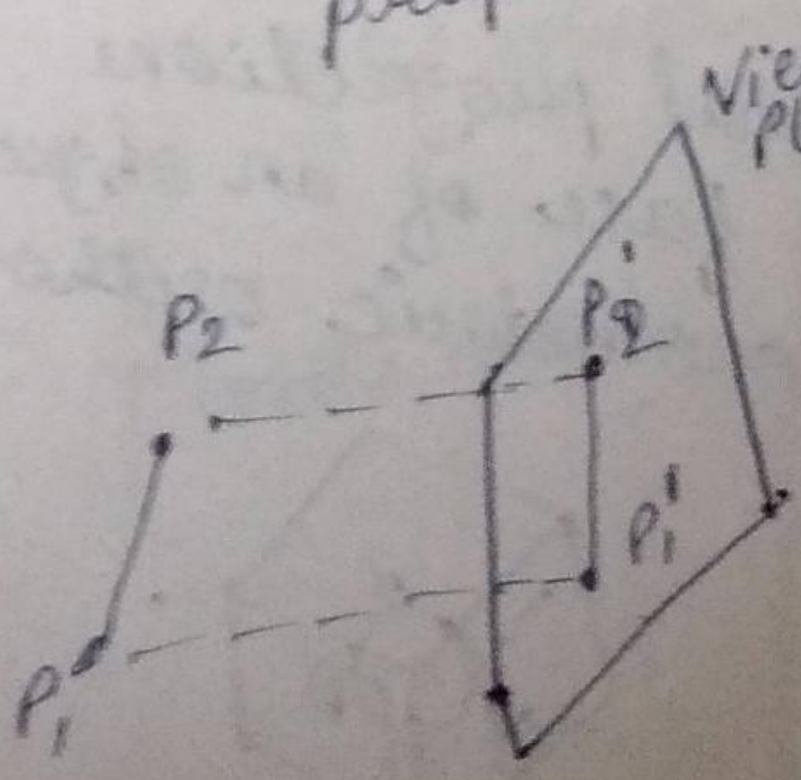
→ In a parallel projection, coordinate positions are transferred to the view plane along parallel lines.

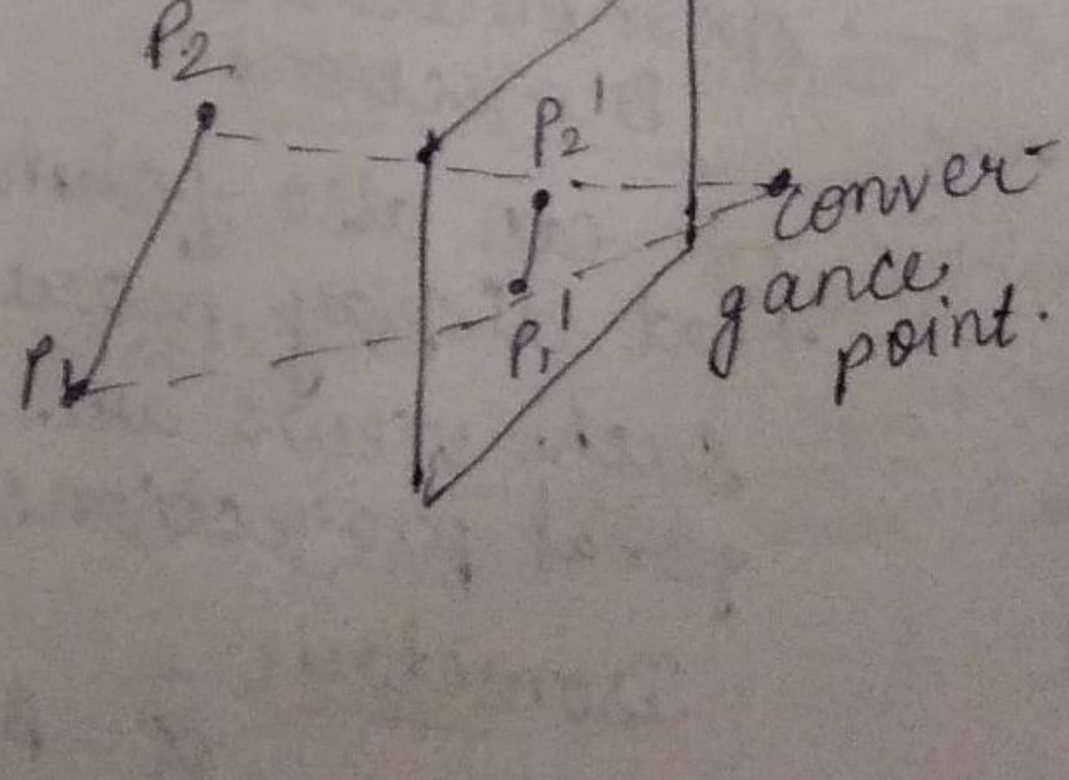
→ There are two general methods for obtaining a parallel-projection view of an object. We can project along lines that are ⊥ar to the view plane, or we can project at an oblique angle to the view plane.

→ Perspective projection:

For a perspective projection, object positions are transformed to projection coordinates along lines that converge to a point behind the view plane.

— Unlike a parallel projections, a perspective projection does not preserve relative proportion of objects.





→ Orthogonal Projections:

— A transformation of object descriptions to a view plane along lines that are all parallel to the view-plane normal vector N is called an orthogonal projection.

— Orthogonal projections are most often used to produce the front, side, and top views of an object.

→ Elevation and Plan View:

Front, side and rear orthogonal projections of an object are called elevations and a top orthogonal projection is called a plan view.

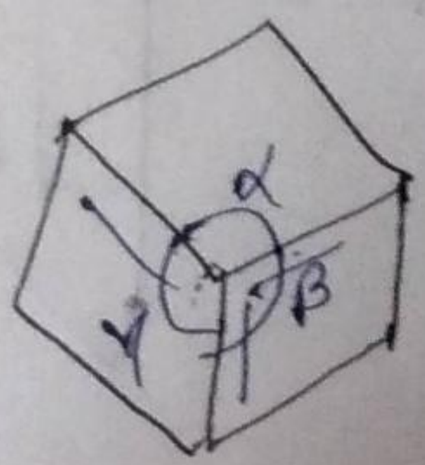
Parallel	Perspective
The distance b/w the COP (centre of projection) and projection plane is infinite.	The distance b/w the COP and projection plane is finite.

→ Axonometric and Isometric Orthogonal Projections:-

We can also form orthogonal projections that display more than one face of an object. Such views are called axonometric orthogonal projections.

Isometric

$\alpha = \beta = \gamma$



Dimetric

$\alpha = \beta \neq \gamma$ or $\alpha \neq \beta = \gamma$

Trimetric

$\alpha \neq \beta \neq \gamma$.

→ Normalization Transformation for an Orthogonal Projection:

- Using an orthogonal transfer of coordinate positions onto the view plane we obtain the projected position of any spatial point (x, y, z) as simply (x, y).
- Thus, once we have established the limits for the view volume, coordinate descriptions inside this rectangular parallelpiped are the projection coordinates, and they can be mapped into a normalized view volume without any further projection processing.
- To illustrate the normalization transformation, we assume that the orthogonal-projection view volume is to be mapped into the symmetric normalization cube within a left-handed reference frame.
- The normalization transformation for the orthogonal view volume is:

$$M_{ortho,norm} = \begin{bmatrix} \frac{2}{x_{wmax} - x_{wmin}} & 0 & 0 & -\frac{x_{wmax} + x_{wmin}}{x_{wmax} - x_{wmin}} \\ 0 & \frac{2}{y_{wmax} - y_{wmin}} & 0 & -\frac{y_{wmax} + y_{wmin}}{y_{wmax} - y_{wmin}} \\ 0 & 0 & \frac{-2}{z_{near} - z_{far}} & \frac{z_{near} + z_{far}}{z_{near} - z_{far}} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

* Oblique Projection :

— An oblique projection is obtained by projecting points along parallel lines that are not perpendicular to the projection plane.

oblique

Cavalier proj.

— the dirⁿ of projection makes a 45° angle with the view plane.

— All lines ⊥ar to the projection plane are projected with no change in length.

Cabinet projections.

— the dirⁿ of projection make a 63.4° angle with the view plane.

— All lines ⊥ʳ to the projection plane are projected at half their length.

$$\text{Mobique} = \begin{bmatrix} 1 & 0 & -\frac{V_{px}}{V_{pz}} & z_{vp}\frac{V_{px}}{V_{pz}} \\ 0 & 1 & -\frac{V_{py}}{V_{pz}} & z_{vp}\frac{V_{py}}{V_{pz}} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

→ Normalization transformation for an oblique Parallel Projection.

— Because the oblique parallel-projection equations convert object decuptions to Orthogonal-coordinate positions, we can apply the normalization procedures following this transformation.

— The Oblique view volume has been converted to a rectangular parallelepiped.

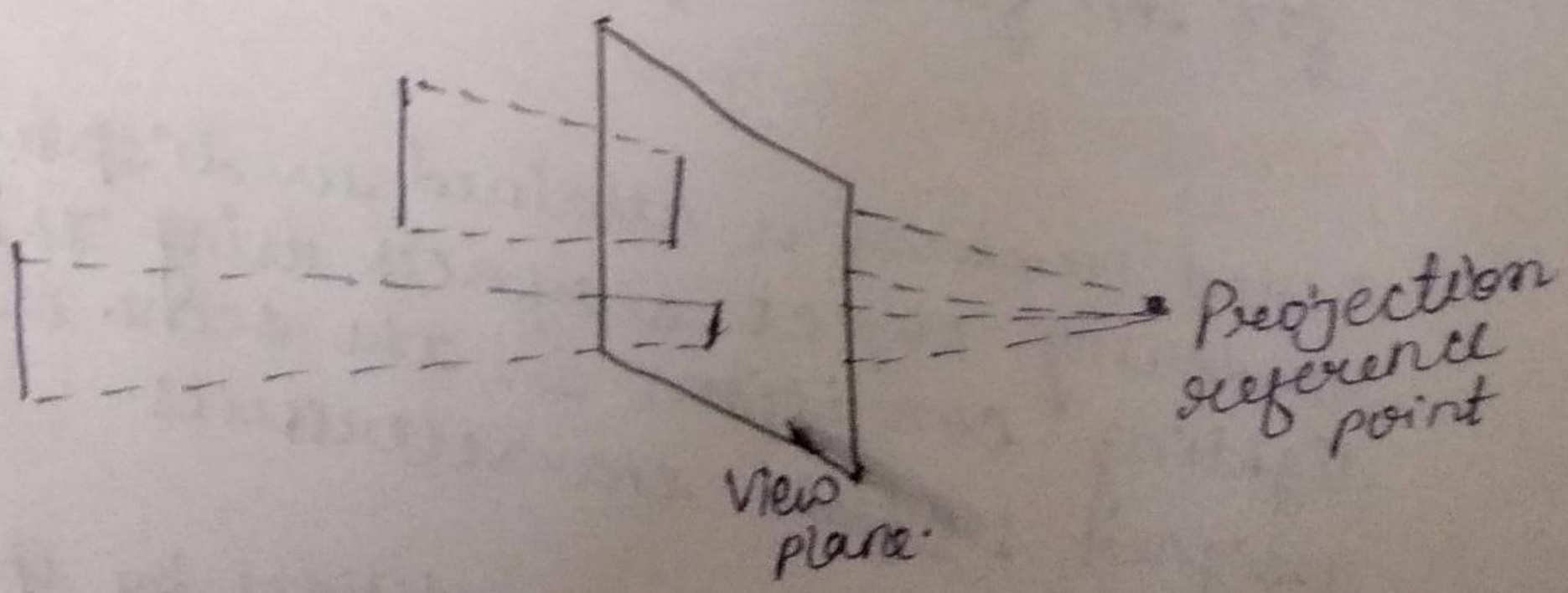
Mobique, norm = Morhe, norm · Mobique

→ Perspective - Projection Transformation Coordinates :—

— We can sometimes select the projection reference point as another viewing parameter in a graphics package, but some systems place this Convergence point at a fixed position, such as at the view point.

— We can write equations describing coordinate positions along this perspective - projection line in parametric form as:

$$x' = x - (x - x_{prp})u$$
$$y' = y - (y - y_{prp})u \qquad 0 \leq u \leq 1$$
$$z' = z - (z - z_{prp})u$$



(iii) Depth Cueing

(iv) Identifying visible Lines and surfaces:

→ We can also clarify depth relationships in a wire-frame display using techniques other than depth cueing.

→ One approach is simply to highlight the visible lines or to display them in a different color.

→ Another techni, we could remove the non-visible lines from the display.

→ When a realistic view of a scene is to be produced, back parts of the objects are completely eliminated so that only the visible surfaces are displayed

→ In this case, surface-rendering procedures are applied so that ~~the~~ screen pixels contain only next ~~parameter~~, patterns for the front surfaces.

(v) Surface Rendering :-

- Added realism is attained in displays by rendering object surfaces using the lighting conditions in the scene and the assigned surface characteristics.

- We set the lighting conditions by specifying the colour and location of the light sources, and we ~~get~~ can also get background illumination effects.

(vi) Exploded and cutaway views :-

- Many graphics packages allow objects to be defined as hierarchical structures, so that internal details can be stored.

- Exploaded and cutaway views of such objects can then be used to show the internal structure and relationship of the object parts.

(vii) three-dimensional and stereoscopic viewing :

- Three-dimensional views can be obtained by reflecting a raster image from a validation, flexible ~~mirror~~ mirror.

- Stereoscopic devices present two views of a ~~scene~~ scene : one for the left eye and the other for the ~~right~~ right eye

* Important terms in Perspective Projections:

- View plane: It is an area of world coordinate system which is projected into viewing plane.

- center of Projection: It is the location of the eye on which projected light rays converge.

- Projectors: It is also called a projection vector. These are rays start from the object scene and are used to create an image of the object on viewing or view plane.

→ It ~~produces~~ introduces several anomalies due to these object shape and appearance gets affected.

- Perspective foreshortening: The size of the object will be small of its distance from the center of projection increases.

- Vanishing Point: All lines appear to meet at some point in the view plane.

- Distortion of Lines: A range lies in front of the viewer to back of viewer is appearing to six rollers.

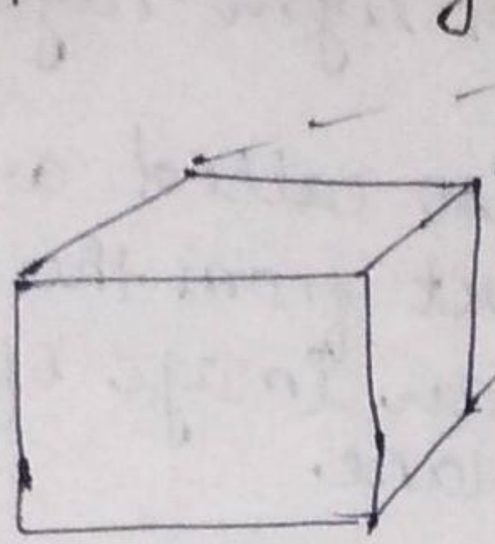
* Vanishing points:

- The point at which a set of projected parallel lines appears to converge is called a vanishing point.

- It is the point where all lines will appear to meet. There can be one point, two point and three point perspective.

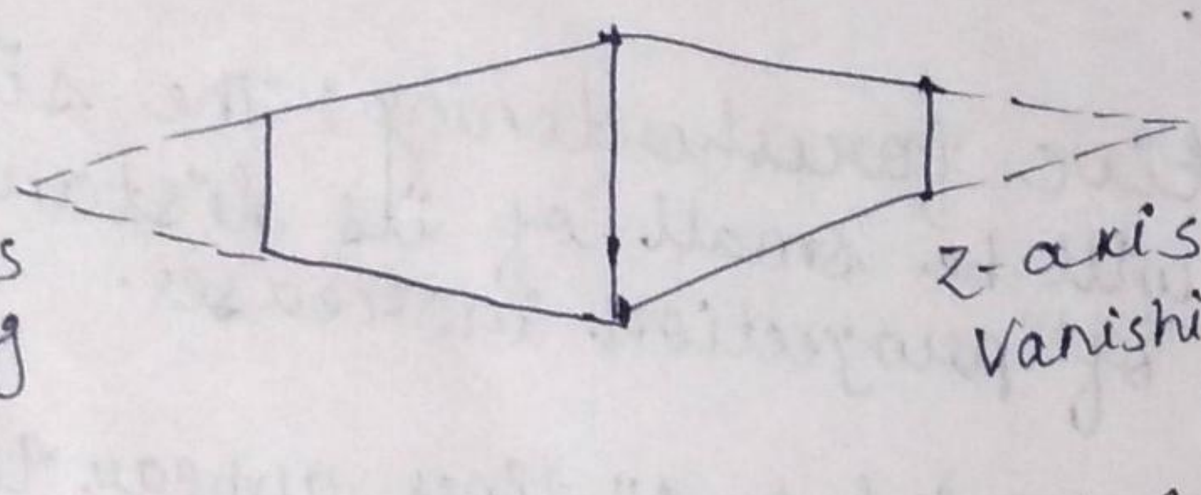
- There is an illusion that certain sets of ll lines (that are not ll to the view plane) appear to meet at some point on the view plane.

→ One principal vanishing point projection :-

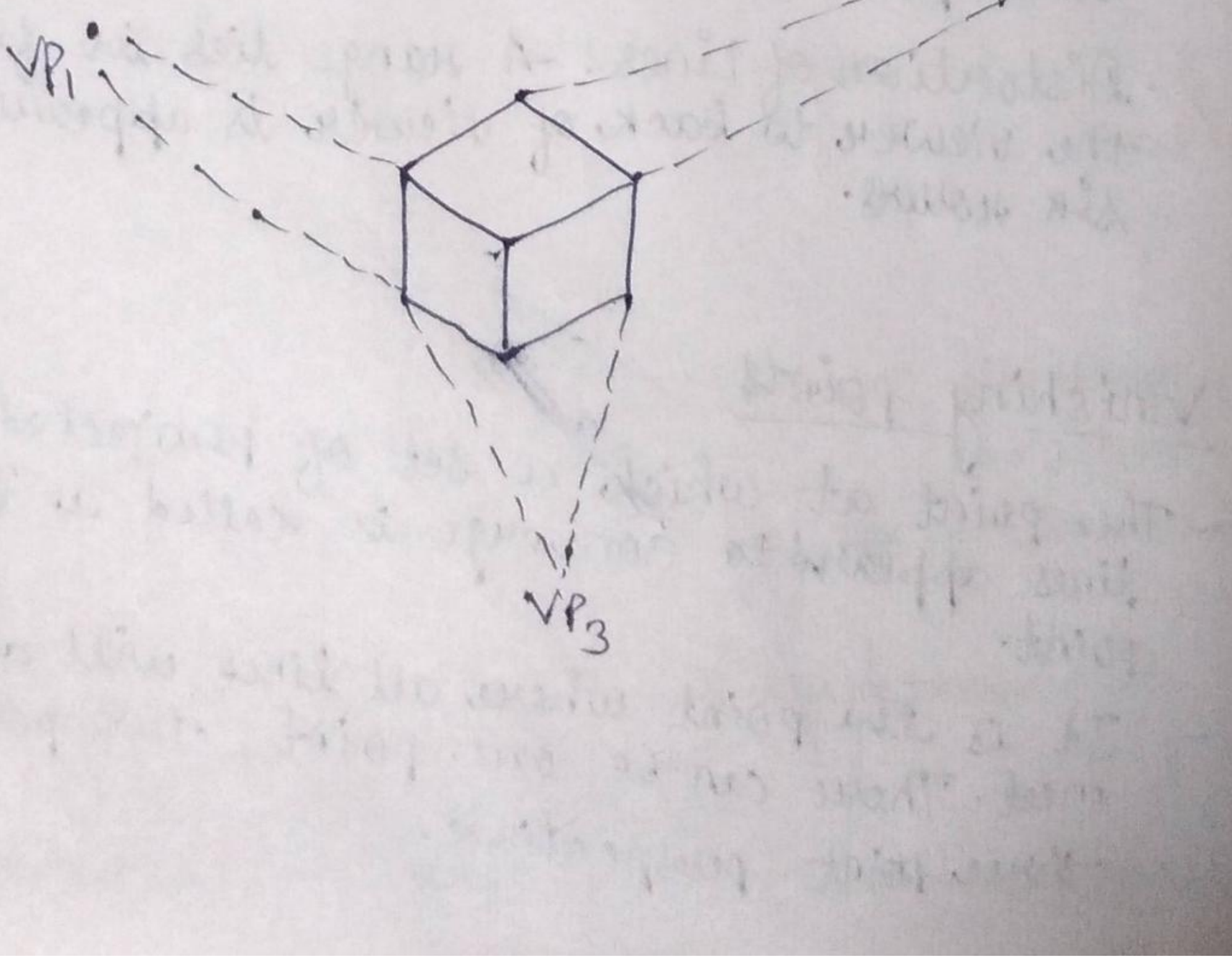


viewplane is ll to XY-plane and intersects z-axis only.

→ Two principal vanishing point projection



- Three principal vanishing point intersection



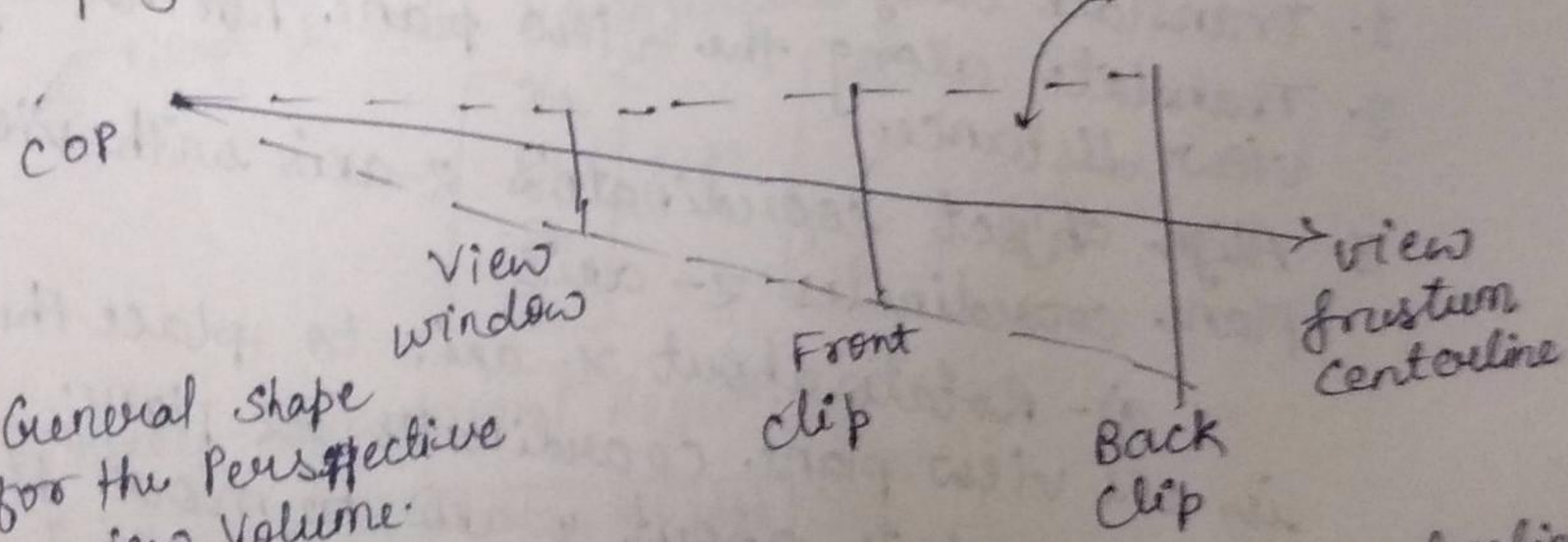
→ View Volume:

- The view volume bounds that portion of the 3D space that is to be clipped out and projected onto the view plane.

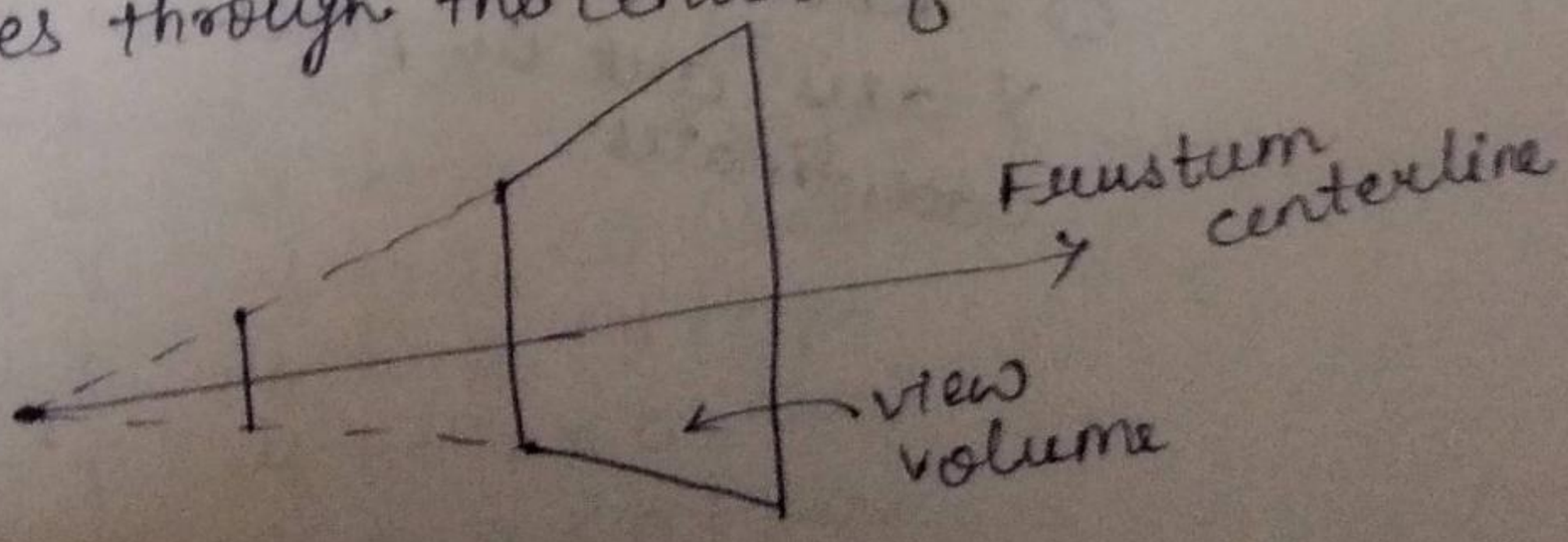
- View Volume for Perspective Projection
  - Its shape is semi-infinite pyramid with apex at the view point and edge passing through the corners of the window.

- View Volume for Parallel Projection:
  - It's shape is an infinite parallelepiped with sides parallel to the direction of projection.

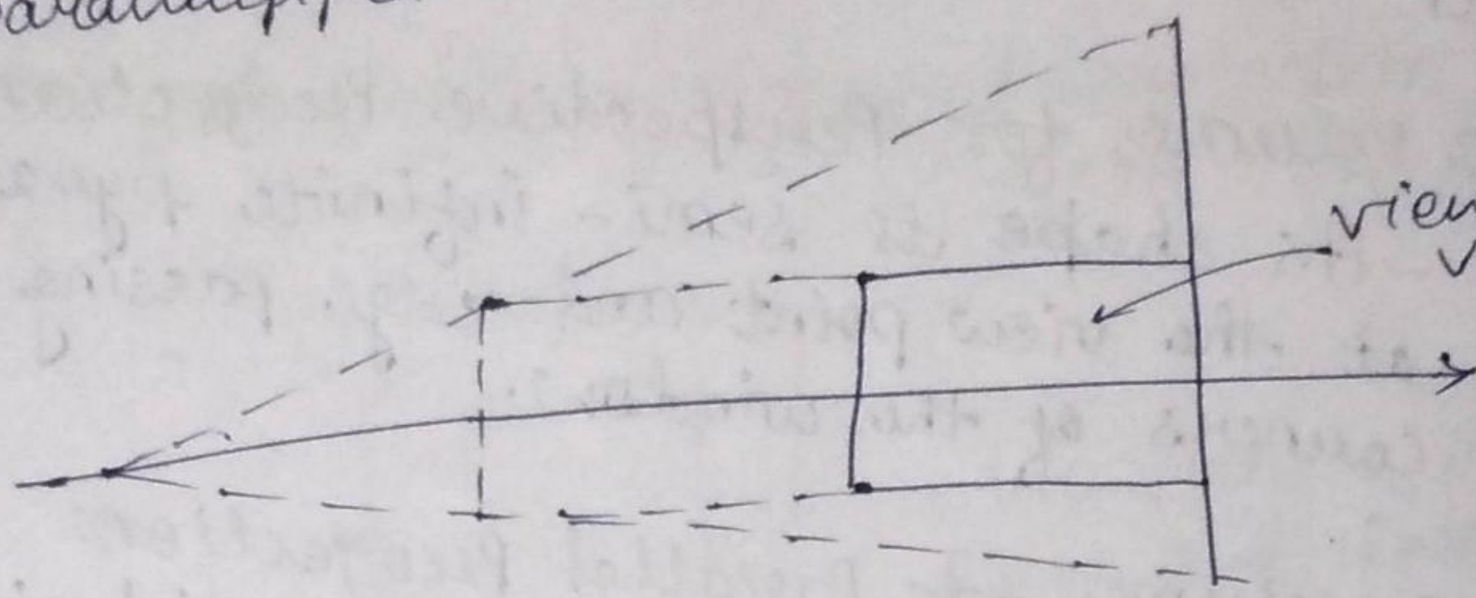
→ Producing a Canonical view volume for a perspective projection.



Step 1:- Shear the view volume so that centerline of the frustum is perpendicular to the view plane and passes through the center of the view window.



step 2: scale view volume inversely proportional to the distance from the view window, so that shape of view volume becomes ~~not~~ rectangular parallelipiped.



* Converting object coordinates to view plane coordinates :-

steps:

1. Translate origin to view reference point (VRP).
2. Translate along the view plane normal by view distance.
3. Align object coordinate's z-axis with view plane coordinates z-axis

   a) - Rotate about x-axis to place the line in the view plane coordinates xz-plane.

   b) - Rotate about y-axis to move the z-axis to its proper position.

   c) - Rotate about the z-axis until x and y axis are in place in view plane coordinates.

** Three-Dimensional clipping Algorithms:

- For the symmetric cube, the equations for the three-dimensional clipping planes are:

$$x_{wmin} = -1, \quad x_{wmax} = 1$$
$$y_{wmin} = -1, \quad y_{wmax} = 1$$
$$z_{wmin} = -1, \quad z_{wmax} = 1.$$

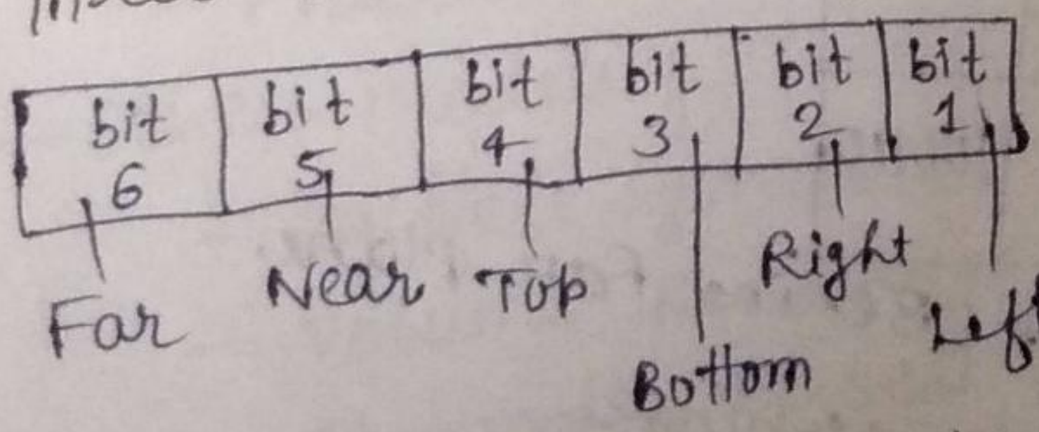
→. Clipping in 3D homogeneous Coordinates

- Computer-graphics libraries process spatial positions as 4-D homogeneous coordinates so that all transformations can be represented as 4×4 matrices.

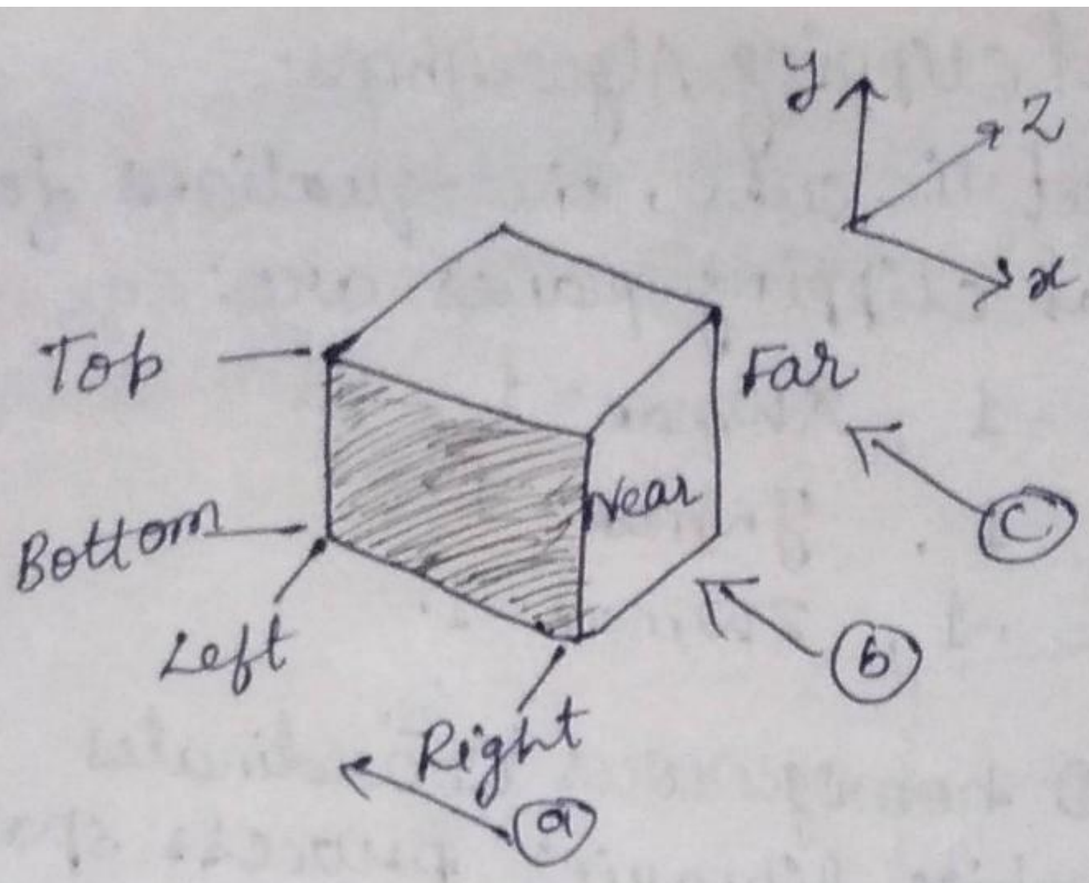
- After a position has passed through the geometric viewing, and projection transformations, it is now in the homogeneous form.

$$\begin{bmatrix} x_h \\ y_h \\ z_h \\ h \end{bmatrix} = M \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

- Three-dimensional Region Codes!



bit 1 = 1 if $h + x_h < 0$ (left)
bit 2 = 1 if $h - x_h < 0$ (right)
bit 3 = 1 if $h + y_h < 0$ (bottom)
bit 4 = 1 if $h - y_h < 0$ (top)
bit 5 = 1 if $h + z_h < 0$ (near)
bit 6 = 1 if $h - z_h < 0$ (far).



011001	011000	011010
010001	010000	010010
010101	010100	010110

(a) Region Codes In Front of Near Plane

001001	001000	001010
000001	000000	000010
000101	000100	000110

(b) Region codes b/w Near & Far Planes

101001	101000	101010
100001	100000	100010
100101	100100	100110

(c) Region Codes Behind Far Plane

** 3D Point and Line Clipping:-

Equations for three-dimensional line segments are conveniently expressed in parametric form, and the clipping methods of Cyrus-Beck or Liang-Barsky can be extended to 3-D scenes.

For a line segment with endpoints $P_1 = (x_{h1}, y_{h1}, z_{h1}, h_1)$ and $P_2 = (x_{h2}, y_{h2}, z_{h2}, h_2)$. We can write the parametric equation describing any point position along the line as.

$$P = P_1 + (P_2 - P_1)u \quad 0 \le u \le 1.$$

→ 3D ~~Poly~~ Polygon clipping.

** 3-D Object Representation :—

General Modeling Techniques :

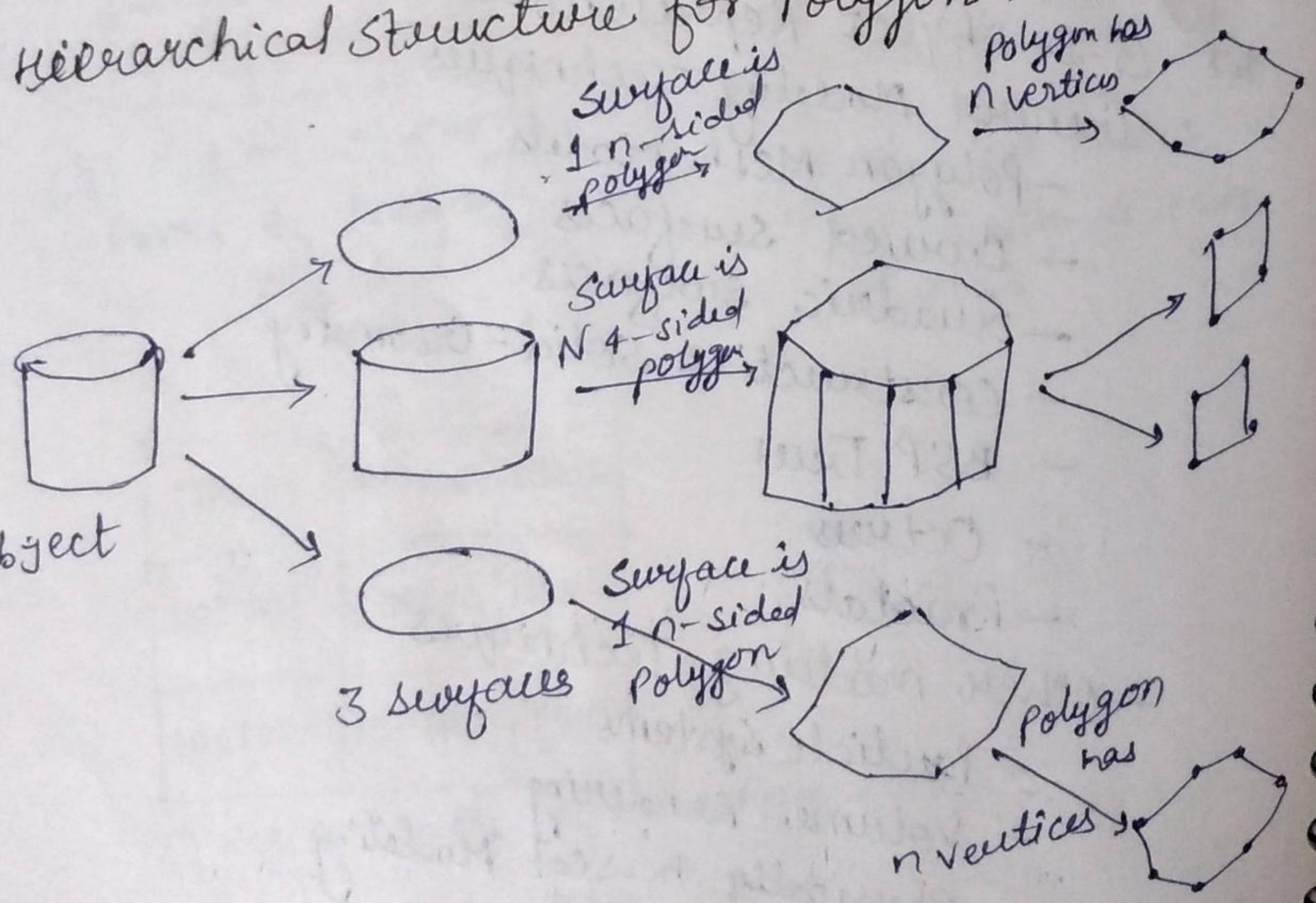
- Polygon Mesh Models
- Curved surfaces
- Quadric surfaces
- Constructive Solid-Geometry
- BSP Trees
- Octrees
- Fractals

- other modeling Techniques
  - Particle Systems
  - Volume Rendering
  - Physically based Modeling

** Polygon Mesh Models

- Most commonly used method for polyhedrons objects.
- Object description is stored as sets of surface polygons.
- Speeds up the surface rendering and display of objects.
- Preview of scene in wireframe representation.

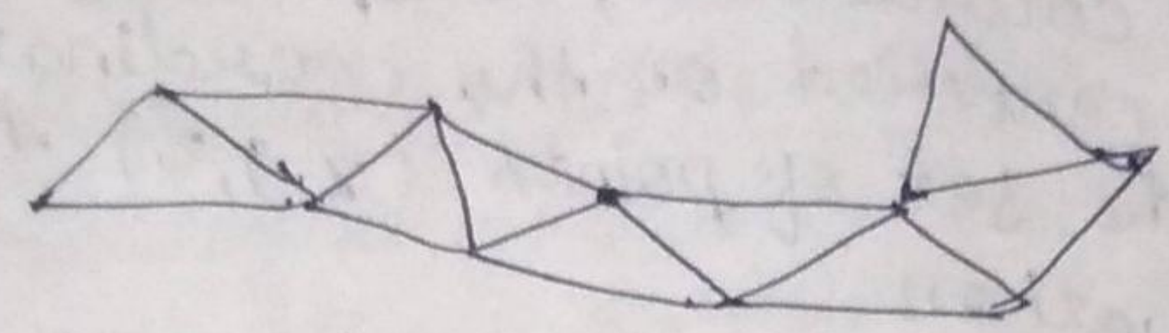
→ Hierarchical Structure for Polygon Mesh Models.



→ Polygon Meshes

- Useful to increase the speed of polygon rendering
- Also helps to decrease the size of the database

Triangular Mesh



— For 'n' vertices, there are 'n-2' connected triangles.
— Major advantage is that they are always flat.

Quadrilateral Mesh

— For n by n array of vertices, a mesh of (n-1) by (m-1) quadrilaterals is generated.

* Curved surfaces:

— Equations for objects with curved boundaries can be expressed in either a parametric or a nonparametric form.

— The various objects that are often useful in graphics applications include quadric surfaces, superquadrics, polynomial and exponential functions, and spline surfaces.

* Quadric Surfaces:

— Quadric surfaces, particularly spheres and ellipsoids are common elements of graphics scenes, and routines for generating these surfaces are often available in graphics packages.

— It can be produced with rational spline representations.

→ Quadric Surfaces : Sphere

In cartesian coordinates, a spherical surface with radius r centered on the coordinate origin is defined as the set of points (x, y, z) that satisfy the equation

$$x^2 + y^2 + z^2 = r^2$$

→ Quadric surfaces : Ellipsoid

An ellipsoidal surface can be described as an extension of a spherical surface where the radii in three mutually perpendicular directions can have different values.

The cartesian representation for points over the surface of an ellipsoid centered on the origin is :

$$\left(\frac{x}{r_x}\right)^2 + \left(\frac{y}{r_y}\right)^2 + \left(\frac{z}{r_z}\right)^2 = 1$$

** Superquadrics :-

— The class of objects called Superquadrics is a generalization of the quadric representations.

— Superquadrics are formed by incorporating additional parameters into the quadric equations to provide increased flexibility for adjusting object shapes.

— One additional parameter is added to curve eqn, and two additional parameters are used in surface equations.

Curves and Surfaces

Superquadrics : superellipse

One way to do this is to write the Cartesian superellipse equation in the form

$$\left(\frac{x}{r_x}\right)^{2/s} + \left(\frac{y}{r_s}\right)^{2/s} = 1. \quad \text{—①}$$

when $s = 1$, we have an ordinary ellipse

→ Corresponding parametric eqn for the superellipse of eqn ① can be expressed as

$$x = r_x \cos^s\theta$$
$$y = r_y \sin^s\theta$$

→ : superellipsoid

A cartesian representation for a superellipsoid is obtained from the equation for an ellipsoid by incorporating two exponent parameters as follows:

$$\left[\left(\frac{x}{r_x}\right)^{2/s_2} + \left(\frac{y}{r_y}\right)^{2/s_2}\right]^{s_2/s_1} + \left(\frac{z}{r_z}\right)^{2/s_1} = 1$$

Curves and surfaces

Bazier Curves & surfaces

Representation of Curves

Brute force Approach

→ A curve can be approximated by a finite number of short straight line segments.

→ To get a better approximation we ~~can~~ can use more segments per unit length.

Some Desirable Properties:

→ Reproducible - the representation should give the same curve every time.

→ Computationally Quick

→ Easy to manipulate, especially important for design purposes.

→ Flexible

→ Easy to combine with other segments of curve.

→ Categories of Curves :-

① Interpolating curves (

- These curves will pass through the points used to describe it.
- The points through which the curve passes are known as knots.

② Approximation curve:-

- An approximating curve will get near to the points w/o necessarily passing through any of them.

Interpolation

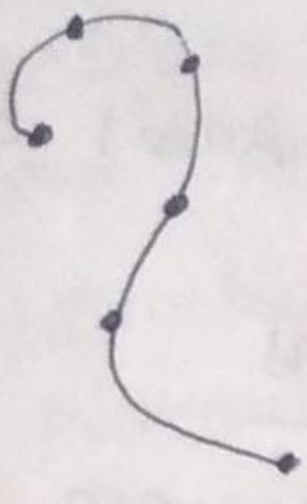
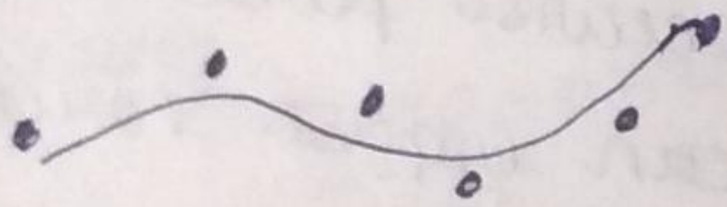


Fig: A set of six control points interpolated with piecewise continuous polynomial section.

Approximation



Explicit vs. Implicit Rep.:

→ The Explicit form is satisfactory when the function is single-valued and the curve has no vertical tangents.

→ The implicitly defined curves require the soln of a non-linear equation for each point and thus numerical procedures have to be employed.

→ Both represented curves are axis-dependent.

* Primitive Polynomial Basis :-

→ In three-dimensional Euclidean space an arbitrary vector is described as a linear combination of three independent vectors.

→ These vectors form a basis for the space and allow the definition of any vector in terms of the basis.

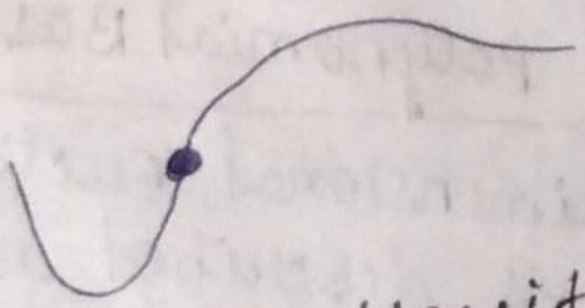
→ Various sets of Basis functions have been used in Computer Graphics for the specification of curves.
- Among these are the Bezier Basis and B-Spline Basis.

→ Parametric continuity conditions
- To ensure a smooth transition from one section of a piecewise parametric curve to the next, we can impose various continuity conditions at the connection points.

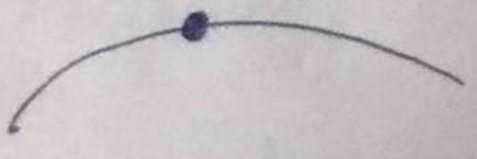
$x = x(u)$, $y = y(u)$, $z = z(u)$, $u_1 \le u \le u_2$

→ Zero order C^0 continuity
↳ last point of first section is same as first point of the second section.

→ First order C^1 continuity.
↳ Tangents of the two sections are the same at their joining point.



→ Second order C^2 continuity!
↳ Both the first & second parametric derivatives of the two curve sections are ~~at~~ the same at the join.



→ Geometric Continuity Conditions

- Another method for joining two successive curve sections is to specify conditions for geometric ~~condition~~ continuity.

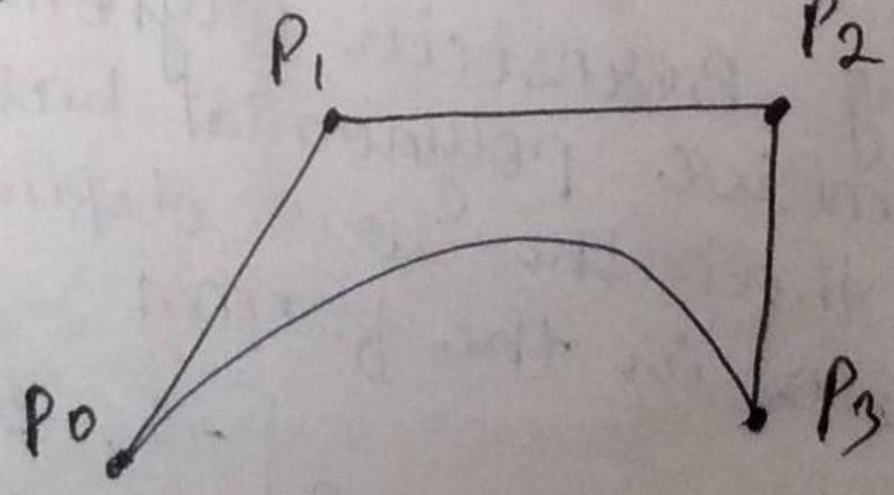
↳ In this case, we require only that the parametric derivatives of the two sections are proportional to each other at their common boundary, instead of requiring equality.

- zero-order geometric continuity
- First " " "
- second " " "

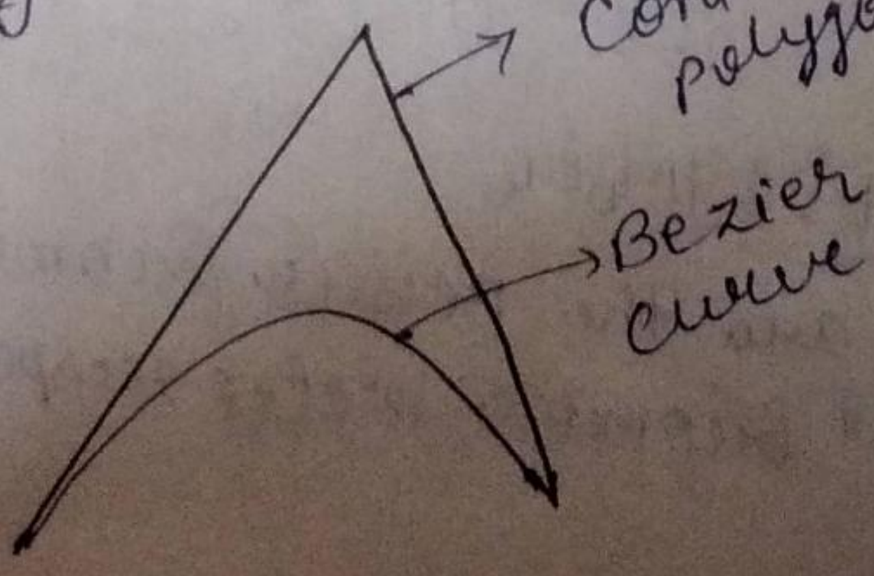
Bezier Curves

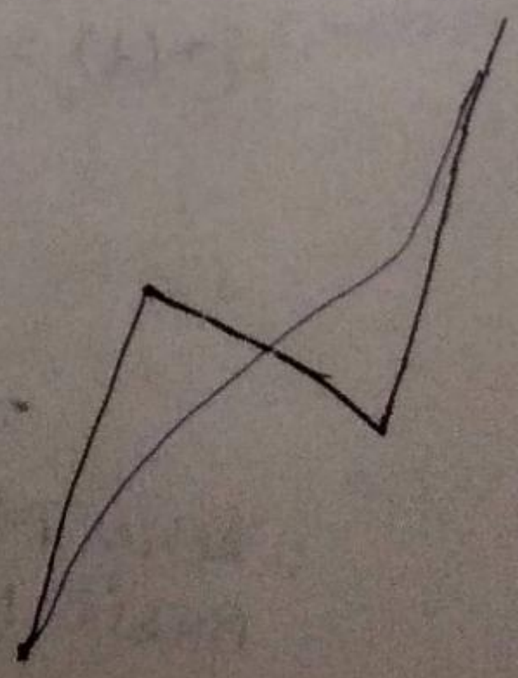
- Bezier functions are a set of polynomials which can be used instead of the primitive polynomial basis and have some useful properties for interactive curve design.

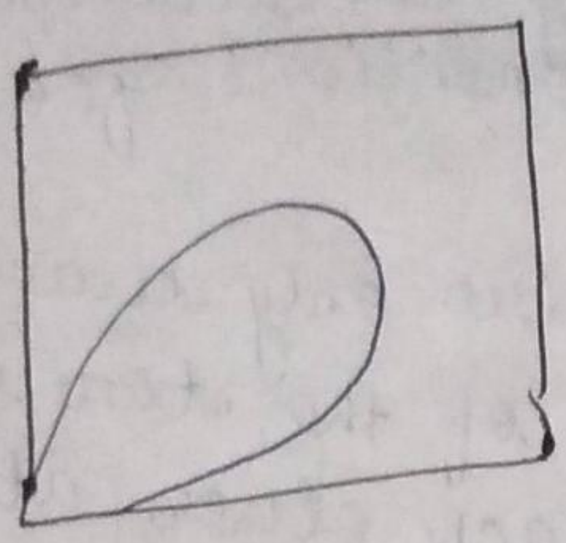
- The cubic Bezier Curve with the four control points P_0, P_1, P_2 and P_3 is illustrated below.

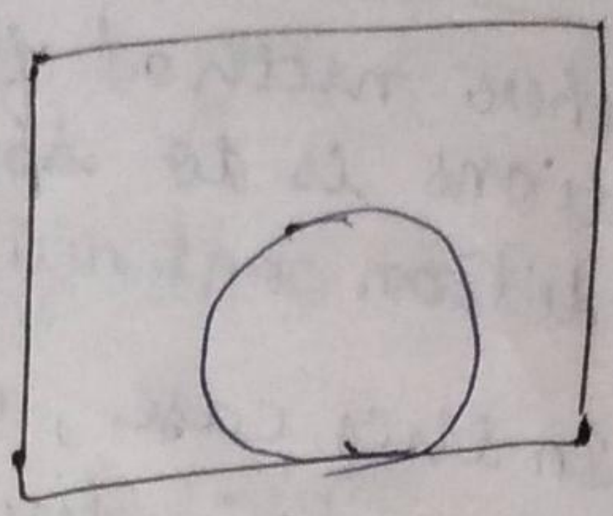


e.g:- 2-D Bezier Curves









imp → The cubic Bezier curve is defined in the range [0,1] in t by:

$$B_3(t) = (1-t)^3 P_0 + 3t(1-t)^2 P_1 + 3t^2(1-t)P_2 + t^3 P_3.$$

→ the cubic Bezier (blending) functions are then:

$(1-t)^3$, $3t(1-t)^2$, $3t^2(1-t)$, t^3.

These are sometimes called the Bernstein polynomials polynomials of order three.

- Expanding Bernstein polynomials in terms of the primitive polynomial basis functions of degree three the curve definition can be written in the form:

$$C(t) = [t^3, t^2, t, 1] \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} P_0 \\ P_1 \\ P_2 \\ P_3 \end{bmatrix}$$

$$= [t^3, t^2, t, 1] M_b G_b.$$

where M_b and G_b are the Bezier Geometric Basis Matrix and Geometric Vector respectively.

→ It is easy to generalize the above to curves of degree n with n+1 control points.

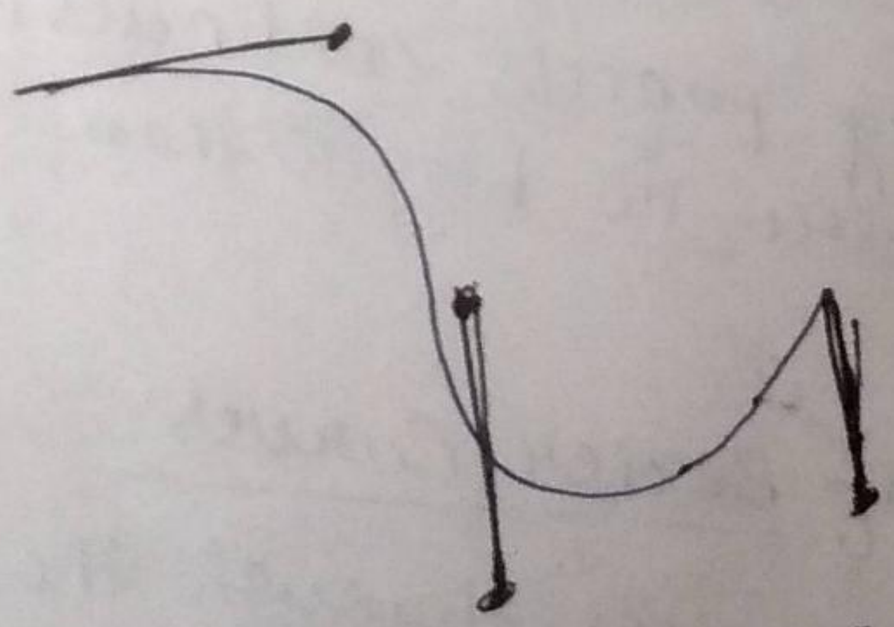
Thus: $$C(t) = \sum_{i=0}^{n} b_{n,i}(t)\, P_i$$

where $b_{n,i}(t) = \binom{n}{i} t^i (1-t)^{n-i}$

→ In practice it is better to use piecewise connected Bezier cubics rather than higher order Bezier curves.

* Piecewise Cubic Bezier Curves :-

- Multiple curve pieces can be joined together to form longer continuous curves.
- The curve is made continuously by by setting the tangents the same at the join.



→ Construction of Bezier curve:-

Recursive Subdivision approach:-

Recursive definition of any point on the surface as series of weighted combinations of control points.

$$P_i^j(t) = (1-t)\, P_i^{j-1}(t) + t\, P_{i+1}^{j-1}(t).$$

for $i = 0, 1, \ldots n$ and $j = 0, 1, 2, \ldots n-i$.

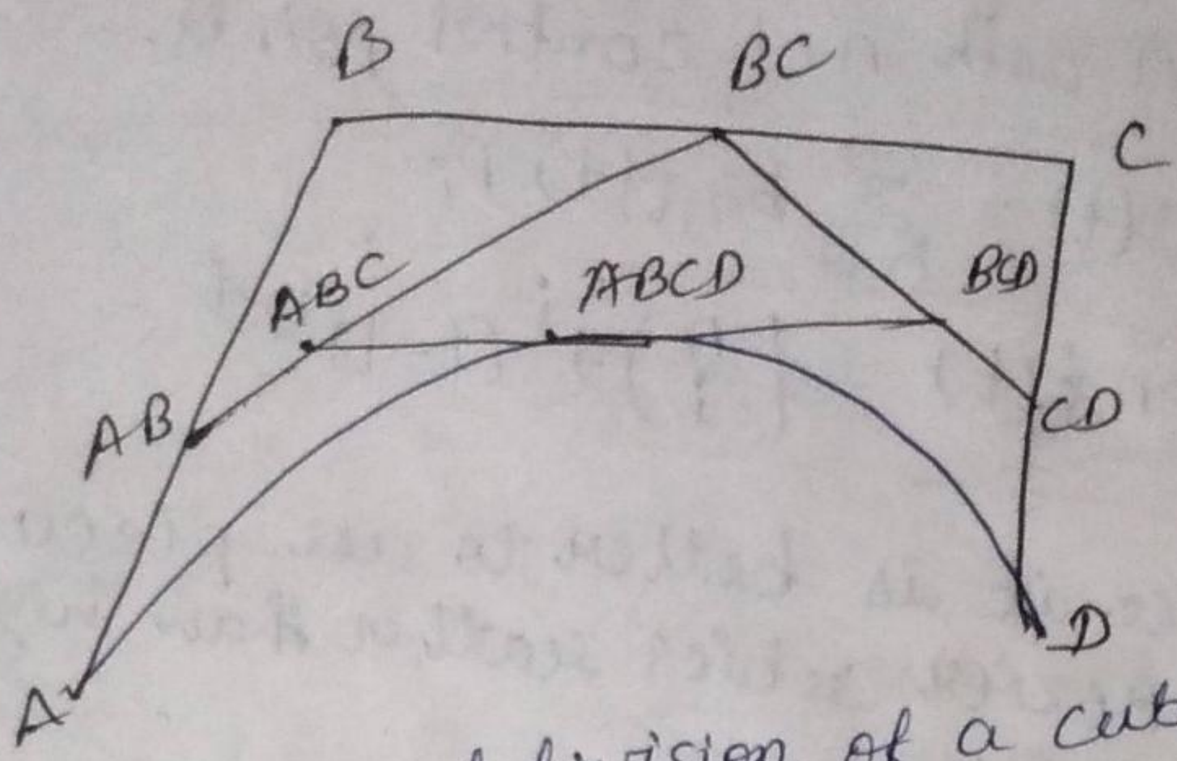


Fig: Subdivision of a cubic Bezier curve

→ The point ABCD is on Bezier curve.

→ Control points for first section of Bezier curve
A, AB, ABC and ABCD

→ Control points for second sections of Bezier curve,
ABCD, BCD, CD and D

This splitting process continues until the sections are no bigger than individual pixel.

** Properties of Bezier Curves

1. The curve passes through the start and finish points of the control polygon defining the curve.

2. The tangent to the curve at $t=0$ lies in the direction of the line joining the first point to the second point. Also the tangent to the curve at $t=1$ is in the direction of the line joining the penultimate point to the last point.

3. Any point on the curve lies inside the convex hull of the control polygon.
4. All control points affect the entire curve.
5. The order of the curve is related to the number of control points.
6. No line can intersect the curve more than twice if the four control points form an open polygon. Thus there can be no loops in the curve and it must be smooth. This is called the Variation Diminishing Property.

* Bezier Surfaces

→ The Bezier curves can be extended to surfaces.
→ We can construct a bicubic Bezier surface from 4*4 grid of control points.

* Spline Curve

→ A spline was originally a flexible strip of metal used by a draughtsman to draw curves.
Three specifications of any spline are :-
- The set of boundary conditions specifying the spline.
- The matrix that characterizes the spline.
- The set of blending functions that characterizes the spline.

① Interpolation spline
② Approximate spline

Cubic-splines interpolation methods:-

Suppose we have n+1 control points specified with coordinates -

$$P_K = (x_K, y_K, z_K) \quad K = 0, 1, 2, \ldots n.$$

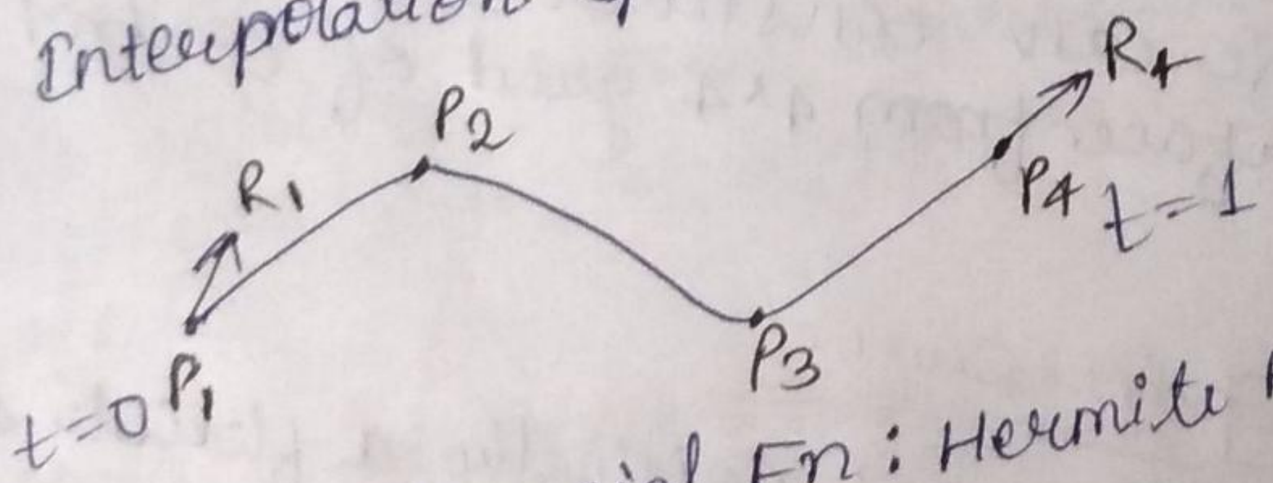
Equations describing cubic polynomials between each pair of control points:

$$x(u) = a_x u^3 + b_x u^2 + c_x u + d_x$$
$$y(u) = a_y u^3 + b_y u^2 + c_y u + d_y$$
$$z(u) = a_z u^3 + b_z u^2 + c_z u + d_z$$

Given a set of "n+1" control points, we want to define 'n' different cubic splines that interpolate the control points.

* Hermite Spline

(i) Interpolation spline curve



(ii) Cubic polynomial Fn : Hermite Fn.

(iii) Hermite Fn : P_1, P_4, R_1, R_4

$$Q(t) = [x(t) \quad y(t) \quad z(t)]$$
$$x(t) = a_x t^3 + b_x t^2 + c_x t + d_x$$
$$y(t) = a_y t^3 + b_y t^2 + c_y t + d_y$$
$$z(t) = a_z t^3 + b_z t^2 + c_z t + d_z$$
$$Q(t) = [t^3 \; t^2 \; t \; 1] \cdot \begin{bmatrix} a_x & a_y & a_z \\ b_x & b_y & b_z \\ c_x & c_y & c_z \\ d_x & d_y & d_z \end{bmatrix}$$

$Q(t) = T \cdot C$ $\quad M$ = basis matrix

$C = M \cdot G_R$ $\quad G_R$ = Geometric vector matrix

$Q(t) = T \cdot M_H \cdot G_H$

$= [t^3\ t^2\ t\ 1] \cdot M_H \cdot G_H$

$Q_x(t) = P_1(t) = [0\ 0\ 0\ 1] \cdot M_H \cdot G_{Hx}$
$t=0$

$Q_x(t) = P_4(t) = [1\ 1\ 1\ 1] \cdot M_H \cdot G_{Hx}$
$t=1$

$Q'_x(t) = R_1(t) = [0\ 0\ 1\ 0] \cdot M_H \cdot G_{Hx}$
$t=0$

$Q'_x(t) = R_4(t) = [3\ 2\ 1\ 0] \cdot M_H \cdot G_{Hx}$
$t=1$

$$\begin{bmatrix} P_1 \\ P_4 \\ R_1 \\ R_4 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 3 & 2 & 1 & 0 \end{bmatrix} \cdot M_H \cdot G_{Hx}$$

$$M_H G_{Hx} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 3 & 2 & 1 & 0 \end{bmatrix}^{-1} \cdot \begin{bmatrix} P_1 \\ P_4 \\ R_1 \\ R_4 \end{bmatrix}$$

$$M_H = \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}, \quad G_{Hx} = \begin{bmatrix} P_1 \\ P_4 \\ R_1 \\ R_4 \end{bmatrix}$$

$$Q(t) = [t^3\ t^2\ t\ 1] \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} P_1 \\ P_2 \\ R_1 \\ R_4 \end{bmatrix}$$

$= (2t^3 - 3t^2 + 1)P_1 + (-2t^3 + 3t^2)P_2 + (t^3 - 2t^2 + t)R_1$
$+ (t^3 - t^2)R_4$

H_1 H_2 H_3 H_4

Rewrite fun : H_1, H_2, H_3, H_4

$$O(t) = H_1 \cdot P_1 + H_2 \cdot P_2 + H_3 \cdot R_1 + H_4 \cdot R_4$$

** ~~Bezier Spline Curve~~ :

① ~~Approximate~~ spline curve

② Bernstein Polynomial func.

③ All control points use - Bez eq

$$O(u) = \sum_{i=0}^{n} P_i * B_{i,n}$$

P_i = Position Vector

$n+1 = 4$

$B_{i,n}$ = Bernstein, Bezier Fun.

B-Spline Curve

↓

Basis

B-spline curve is used to remove drawbacks of Bazier curve.

Drawback of Bazier Curve -

① polynomial degree is decided by control points

C.P = 5

P.O = 5-1 = 4 (x^4)

② Blending Fn.

↓

non zero

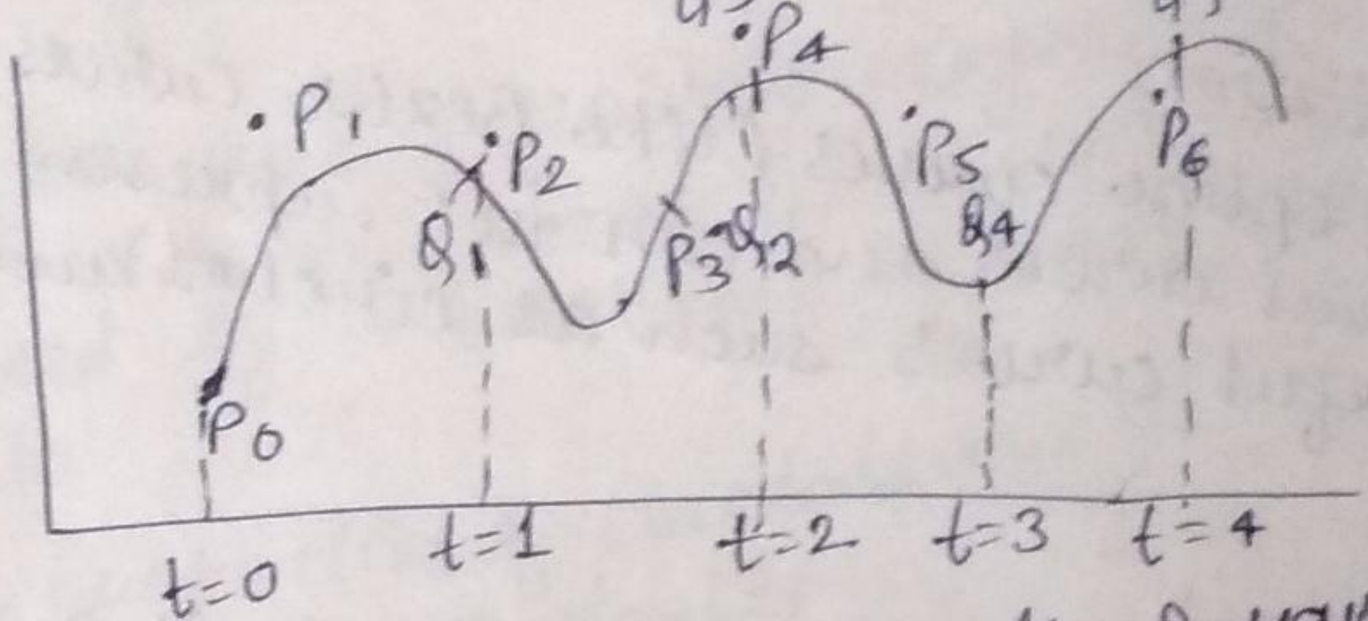
→ global control

↓

means if one point position changes then whole curve change

① Approximate Spline Curve
② Local control point
③ blending fn. not necessary non zero
↳ it can be 0 also

④ Control point = 15 (x^{14})
but we can specify the degree of polygon
specifying order of curve(K).



Control point = 7
K = 3
Polydeg = 2.
n-K+2
= ⑤

Segment	Control Points	Parameter
Q_1	$P_0 P_1 P_2$	$t_0=0$ $t_1=1$
Q_2	$P_1 P_2 P_3$	$t_1=1$, $t_2=2$
Q_3	$P_2 P_3 P_4$	$t_2=2$, $t_3=3$.
Q_4	$P_3 P_4 P_5$	$t_3=3$, $t_4=4$
Q_5	$P_4 P_5 P_6$	$t_4=4$, $t_5=5$

$$\boxed{Q(u) = \sum_{i=0}^{n} P_i \cdot N_{i,K}(u)}$$ → B-spline bases fn.

$$N_{i,K}(u) = \frac{(u-x_i)N_{i,K-1}(u)}{x_{i+K-1}-x_i} + \frac{(x_{i+K}-u)N_{i+1,K-1}(u)}{x_{i+K}-x_{i+1}}$$

x_i $(0 \le i \le n+K)$

$x_i = 0$ if $i < K$
$x_i = i-K+1$ if $K \le i \le n$
$x_i = n-K+2$ if $i > n$

Cond^n for stop:

$N_{i,K}(u) = 1$ if $x_i \leq u \leq x_{i+1}$

$= 0$

Types of B-spline

① Open B-spline
② Clamped B-spline
③ closed B-spline

Limitation

B-spline curves (like Bezier curves) are polynomial curves and can not represent many useful curves such as circles and ellipses.

Fractals

- Modeling of complex objects is a difficult process.

→ ~~In particular, geometric modeling of complex forms or mathematical mo~~

- Natural objects such as coastlines, mountains cloud formations, trees etc. constitute an important class of complex objects.

- In order to represent these complex shapes, the traditional methods like the Euclidean geometry are unable to describe elegantly such complex objects.

- The family of complex shapes described by fractals.

Euclidean	Fractal
- Traditional - Based on charac. size & scale - Suits for man-made objects - Described by a non-recursive formula	- Modern monsters - No specific size and scale - Appropriate for natural shapes - Described by algorithm or recursive formula.

→ Fractatals are infinitely magnifiable irregular objects with fractional dimension which can be produced by a small set of instructions and data.

→ Imp. properties of fractals are :-
- Self-similarity
- Fractional Dimensions
- Formation by iteration.

Topological dimension

↳ all the geometric objects residing in Euclidean space has integer dimension. Such a dimension is alternatively also known as topological dimension.

Fractal dimension

- In fractal geometry, dimension of an object is not an integer rather a fraction like 1.53, 2.71 etc.
- Simply this implies that objects are possible where dimension is b/w 1 and 2.

In Euclidean dimension D
in fractal $D = \frac{\log(N)}{\log(r)}$.

* Visible-surface detection

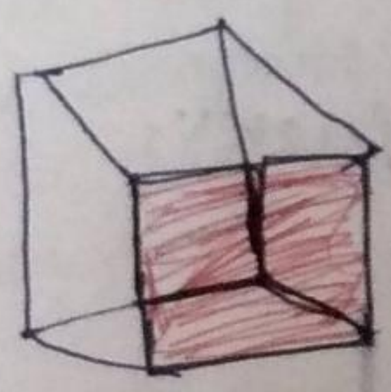
→ It is also referred to as hidden-surface elimination methods.

— In a given set of 3D objects and viewing specification, we wish to determine which lines or surfaces of the objects are visible, so that we can display only the visible lines or surfaces. This process is known as hidden surfaces or hidden line elimination.

— The hidden line or hidden surface algo determines the lines, edges, surfaces or volumes that are visible or invisible

— Two approaches
① Object-space method
② Image-space method

Processing-time
Render-Image
↑ extra info.



object space Method	Image space Method
1) Physical Coordinate	1) Viewport / Projection port coordinates
2) object part → visibility	2) Point to point pixel ↓ Visible
③ Line-display algorithm	③ Line/surfaces algorithm
④ Geometrical relationship among objects	④ Final image concentrate
⑤ Geometrical calculation:- precise → float data precision	⑤ Pixel : 500×500

⑥ Time processing increases	⑥ Less time consuming.
⑦ consider object only one time but pixel can be process many time.	⑦ Consider pixel only once but object may be many times.

* coherence
  ↳ similarity

① object coherence : Seperate object (Non overlapping)

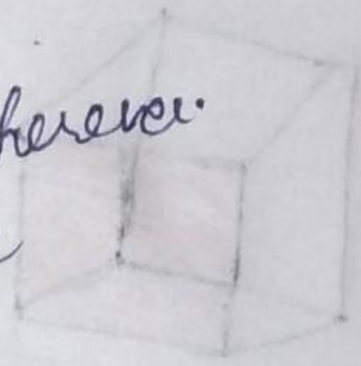
② Face coherence :
surface

③ Edge coherence

④ Area coherence
window

⑤ Implied Edge coherence.

⑥ Depth coherence



Back-face detection

— A fast and simple object-space method for locating the back faces of a polyhedron is based on front-back tests. A point (x, y, z) is behind a polygon surface if

$$Ax + By + Cz + D < 0$$

where A, B, C, and D are the plane parameters for the polygon.

→ When this position is along the line of sight to the surface, we must be looking at the back of the polygon. Therefore, we ~~should~~ could use the viewing position to test for back faces.

* Depth-buffer Method

— A commonly used image-space approach for detecting visible surfaces is the depth-buffer method, which compares surface depth values throughout a scene for each pixel position on the projection plane.

— Each surface of a scene is processed separately, one pixel position at a time, across the surface.

— The algo is usually applied to scenes containing only polygon surfaces, because depth values can be computed very quickly and the method is easy to implement.